NAG Toolbox

nag_fit_glinc_l1sol (e02gb)

1 Purpose

nag_fit_glinc_l1sol (e02gb) calculates an l_1 solution to an over-determined system of linear equations, possibly subject to linear inequality constraints.

2 Syntax

```
[e, x, k, el1n, indx, ifail] = nag_fit_glinc_l1sol(m, e, f, x, mxs, monit, iprint, 'n', n, 'mpl', mpl)
[e, x, k, el1n, indx, ifail] = e02gb(m, e, f, x, mxs, monit, iprint, 'n', n, 'mpl', mpl)
```

3 Description

Given a matrix A with m rows and n columns $(m \geq n)$ and a vector b with m elements, the function calculates an l_1 solution to the over-determined system of equations

$$Ax = b.$$

That is to say, it calculates a vector x, with n elements, which minimizes the l_1-norm (the sum of the absolute values) of the residuals

$$r(x) = \sum_{i=1}^{m} |r_i|,$$

where the residuals r_i are given by

$$r_i = b_i - \sum_{j=1}^{n} a_{ij}x_j, \quad i = 1, 2, \ldots, m.$$

Here a_{ij} is the element in row i and column j of A, b_i is the ith element of b and x_j the jth element of x.

If, in addition, a matrix C with l rows and n columns and a vector d with l elements, are given, the vector x computed by the function is such as to minimize the l_1-norm $r(x)$ subject to the set of inequality constraints $Cx \geq d$.

The matrices A and C need not be of full rank.

Typically in applications to data fitting, data consisting of m points with coordinates (t_i, y_i) is to be approximated by a linear combination of known functions $\phi_i(t)$,

$$\alpha_1\phi_1(t) + \alpha_2\phi_2(t) + \cdots + \alpha_n\phi_n(t),$$

in the l_1-norm, possibly subject to linear inequality constraints on the coefficients α_j of the form $C\alpha \geq d$ where α is the vector of the α_j and C and d are as in the previous paragraph. This is equivalent to finding an l_1 solution to the over-determined system of equations

$$\sum_{j=1}^{n} \phi_j(t_i)\alpha_j = y_i, \quad i = 1, 2, \ldots, m,$$

subject to $C\alpha \geq d$.

Thus if, for each value of i and j, the element a_{ij} of the matrix A above is set equal to the value of $\phi_j(t_i)$ and b_i is equal to y_i and C and d are also supplied to the function, the solution vector x will contain the required values of the α_j. Note that the independent variable t above can, instead, be a

vector of several independent variables (this includes the case where each of ϕ_i is a function of a different variable, or set of variables).

The algorithm follows the Conn–Pietrzykowski approach (see Bartels *et al.* (1978) and Conn and Pietrzykowski (1977)), which is via an exact penalty function

$$g(x) = \gamma r(x) - \sum_{i=1}^{l} \min\bigl(0, c_i^{\mathrm{T}} x - d_i\bigr),$$

where γ is a penalty parameter, c_i^{T} is the ith row of the matrix C, and d_i is the ith element of the vector d. It proceeds in a step-by-step manner much like the simplex method for linear programming but does not move from vertex to vertex and does not require the problem to be cast in a form containing only non-negative unknowns. It uses stable procedures to update an orthogonal factorization of the current set of active equations and constraints.

4 References

Bartels R H, Conn A R and Charalambous C (1976) Minimisation techniques for piecewise Differentiable functions – the l_∞ solution to an overdetermined linear system *Technical Report No. 247, CORR 76/30* Mathematical Sciences Department, The John Hopkins University

Bartels R H, Conn A R and Sinclair J W (1976) A Fortran program for solving overdetermined systems of linear equations in the l_1 Sense *Technical Report No. 236, CORR 76/7* Mathematical Sciences Department, The John Hopkins University

Bartels R H, Conn A R and Sinclair J W (1978) Minimisation techniques for piecewise differentiable functions – the l_1 solution to an overdetermined linear system *SIAM J. Numer. Anal.* **15** 224–241

Conn A R and Pietrzykowski T (1977) A penalty-function method converging directly to a constrained optimum *SIAM J. Numer. Anal.* **14** 348–375

5 Parameters

5.1 Compulsory Input Parameters

1: **m** – INTEGER

The number of equations in the over-determined system, m (i.e., the number of rows of the matrix A).

Constraint: $\mathbf{m} \geq \mathbf{n}$.

2: **e**(lde, **mpl**) – REAL (KIND=nag_wp) array

lde, the first dimension of the array, must satisfy the constraint $lde \geq \mathbf{n}$.

The equation and constraint matrices stored in the following manner.

The first m columns contain the m rows of the matrix A; element $\mathbf{e}(i,j)$ specifying the element a_{ji} in the jth row and ith column of A (the coefficient of the ith unknown in the jth equation), for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$. The next l columns contain the l rows of the constraint matrix C; element $\mathbf{e}(i, j+m)$ containing the element c_{ji} in the jth row and ith column of C (the coefficient of the ith unknown in the jth constraint), for $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, l$.

3: **f**(**mpl**) – REAL (KIND=nag_wp) array

$\mathbf{f}(i)$, for $i = 1, 2, \ldots, m$, must contain b_i (the ith element of the right-hand side vector of the over-determined system of equations) and $\mathbf{f}(m+i)$, for $i = 1, 2, \ldots, l$, must contain d_i (the ith element of the right-hand side vector of the constraints), where l is the number of constraints.

4: **x**(**n**) – REAL (KIND=nag_wp) array

$\mathbf{x}(i)$ must contain an estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If no better initial estimate for $\mathbf{x}(i)$ is available, set $\mathbf{x}(i) = 0.0$.

5: **mxs** – INTEGER

The maximum number of steps to be allowed for the solution of the unconstrained problem. Typically this may be a modest multiple of n. If, on entry, **mxs** is zero or negative, the value returned by nag_machine_integer_max (x02bb) is used.

6: **monit** – SUBROUTINE, supplied by the user.

monit can be used to print out the current values of any selection of its arguments. The frequency with which **monit** is called in nag_fit_glinc_l1sol (e02gb) is controlled by **iprint**.

```
monit(n, x, niter, k, el1n)
```

Input Parameters

1: **n** – INTEGER

The number n of unknowns (the number of columns of the matrix A).

2: **x**(**n**) – REAL (KIND=nag_wp) array

The latest estimate of the unknowns.

3: **niter** – INTEGER

The number of iterations so far carried out.

4: **k** – INTEGER

The total number of equations and constraints which are currently active (i.e., the number of equations with zero residuals plus the number of constraints which are satisfied as equations).

5: **el1n** – REAL (KIND=nag_wp)

The l_1-norm of the current residuals of the over-determined system of equations.

7: **iprint** – INTEGER

The frequency of iteration print out.

iprint > 0
monit is called every **iprint** iterations and at the solution.

iprint $= 0$
Information is printed out at the solution only. Otherwise **monit** is not called (but a dummy function must still be provided).

5.2 Optional Input Parameters

1: **n** – INTEGER

Default: the dimension of the array **x** and the first dimension of the array **e**. (An error is raised if these dimensions are not equal.)

The number of unknowns, n (the number of columns of the matrix A).

Constraint: **n** ≥ 2.

2: **mpl** – INTEGER

Default: the dimension of the array **f** and the second dimension of the array **e**. (An error is raised if these dimensions are not equal.)

$m + l$, where l is the number of constraints (which may be zero).

Constraint: **mpl** $\geq$ **m**.

5.3 Output Parameters

1: **e**$(lde, \mathbf{mpl})$ – REAL (KIND=nag_wp) array

Unchanged, except possibly to the extent of a small multiple of the ***machine precision***. (See Section 9.)

2: **x**$(\mathbf{n})$ – REAL (KIND=nag_wp) array

The latest estimate of the ith unknown, for $i = 1, 2, \ldots, n$. If **ifail** $= 0$ on exit, these are the solution values.

3: **k** – INTEGER

The total number of equations and constraints which are then active (i.e., the number of equations with zero residuals plus the number of constraints which are satisfied as equalities).

4: **el1n** – REAL (KIND=nag_wp)

The l_1-norm (sum of absolute values) of the equation residuals.

5: **indx**$(\mathbf{mpl})$ – INTEGER array

Specifies which columns of **e** relate to the inactive equations and constraints. **indx**(1) up to **indx**$(\mathbf{k})$ number the active columns and **indx**$(\mathbf{k} + 1)$ up to **indx**$(\mathbf{mpl})$ number the inactive columns.

6: **ifail** – INTEGER

ifail $= 0$ unless the function detects an error (see Section 5).

6 Error Indicators and Warnings

Errors or warnings detected by the function:

ifail $= 1$

The constraints cannot all be satisfied simultaneously: they are not compatible with one another. Hence no solution is possible.

ifail $= 2$

The limit imposed by **mxs** has been reached without finding a solution. Consider restarting from the current point by simply calling nag_fit_glinc_l1sol (e02gb) again without changing the arguments.

ifail $= 3$

The function has failed because of numerical difficulties; the problem is too ill-conditioned. Consider rescaling the unknowns.

ifail $= 4$

Constraint: $lde \geq$ **n**.

Constraint: **mpl** $\geq$ **m**.

Constraint: **m** $\geq$ **n**.

Constraint: **n** ≥ 2.

Elements 1 to **m** of one of the first **mpl** columns of the array **e** are all zero – this corresponds to a zero row in either of the matrices A or C.

On entry, iw is too small.

ifail $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

ifail $= -399$

Your licence key may have expired or may not have been installed correctly.

ifail $= -999$

Dynamic memory allocation failed.

7 Accuracy

The method is stable.

8 Further Comments

The effect of m and n on the time and on the number of iterations varies from problem to problem, but typically the number of iterations is a small multiple of n and the total time taken is approximately proportional to mn^2.

Linear dependencies among the rows or columns of A and C are not necessarily a problem to the algorithm. Solutions can be obtained from rank-deficient A and C. However, the algorithm requires that at every step the currently active columns of **e** form a linearly independent set. If this is not the case at any step, small, random perturbations of the order of rounding error are added to the appropriate columns of **e**. Normally this perturbation process will not affect the solution significantly. It does mean, however, that results may not be exactly reproducible.

9 Example

Suppose we wish to approximate in $[0, 1]$ a set of data by a curve of the form

$$y = ax^3 + bx^2 + cx + d$$

which has non-negative slope at the data points. Given points (t_i, y_i) we may form the equations

$$y_i = at_i^3 + bt_i^2 + ct_i + d$$

for $i = 1, 2, \ldots, 6$, for the 6 data points. The requirement of a non-negative slope at the data points demands

$$3at_i^2 + 2bt_i + c \geq 0$$

for each t_i and these form the constraints.

(Note that, for fitting with polynomials, it would usually be advisable to work with the polynomial expressed in Chebyshev series form (see the E02 Chapter Introduction). The power series form is used here for simplicity of exposition.)

9.1 Program Text

```
function e02gb_example

fprintf('e02gb example results\n\n');

n = 4;
m = nag_int(6);
f = zeros(2*m,1);
```

```
e = zeros(n,2*m);
x = zeros(n,1);

% Data to fit by constrained cubic polynomial
f(1:m) = [0, 0.07,  0.07,  0.11,  0.27,  0.68];

% coefficient multipliers for cubic and its derivative
for i = 1:m
  t = 0.2*double(i-1);
  e(1:4,i) = [1,t,t^2,t^3];
  e(1:4,m+i) = [0,1,2*t,3*t^2];
end

mxs    = nag_int(100);
iprint = nag_int(0);
[e, x, k, l1nrm, indx, ifail] = e02gb(m, e, f, x, mxs, @monit, iprint);

function [] = monit(n, x, niter, k, elin)

  fprintf('\n Results at iteration %d\n', niter);
  fprintf('Constrained cubic polynomial:\n\n');
  fprintf('p(t) = %7.4f + %7.4f t + %7.4f t^2 + %7.4f t^3\n\n',x(1:4));
  fprintf('Norm of residuals = %12.5f\n', elin);
```

9.2 Program Results

```
     e02gb example results

 Results at iteration 8
Constrained cubic polynomial:

p(t) =  0.0000 +  0.6943 t + -2.1482 t^2 +  2.1339 t^3

Norm of residuals =      0.00957
```